AD

TECHNICAL REPORT ARCCB-TR-95041

LINEAR TIME ALGORITHM FOR POSITIVE KERNEL SMOOTHING WITH APPLICATION TO NONPARAMETRIC PROBABILITY DENSITY ESTIMATION

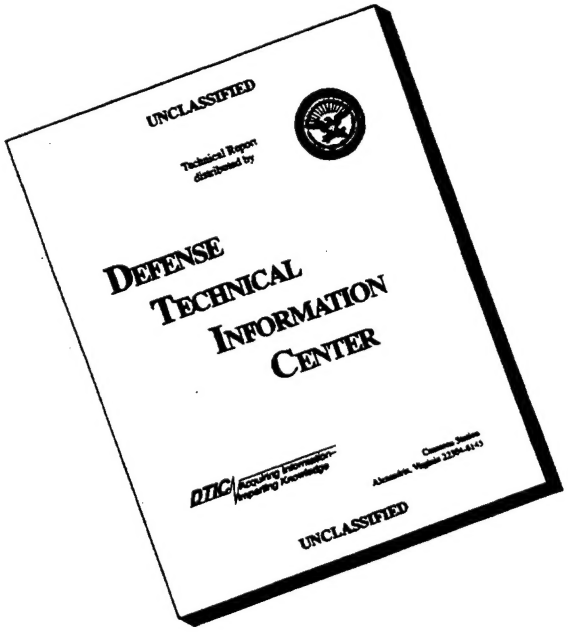
ROYCE W. SOANES

OCTOBER 1995

US ARMY ARMAMENT RESEARCH,
DEVELOPMENT AND ENGINEERING CENTER
CLOSE COMBAT ARMAMENTS CENTER
BENÉT LABORATORIES
WATERVLIET, N.Y. 12189-4050

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The use of trade name(s) and/or manufacturer(s) does not constitute an official indorsement or approval.

DESTRUCTION NOTICE

For classified documents, follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For unclassified, limited documents, destroy by any method that will prevent disclosure of contents or reconstruction of the document.

For unclassified, unlimited documents, destroy when the report is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1995	Final

4. TITLE AND SUBTITLE
LINEAR TIME ALGORITHMS FOR POSITIVE KERNEL SMOOTHING WITH APPLICATION TO NONPARAMETRIC PROBABILITY DENSITY ESTIMATION

5. FUNDING NUMBERS
AMCMS: 6226.24.H180.0
PRON: 956M387

6. AUTHOR(S)
Royce W. Soanes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Benét Laboratories, AMSTA-AR-CCB
Watervliet, NY 12189-4050

8. PERFORMING ORGANIZATION REPORT NUMBER
ARCCB-TR-95041

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Close Combat Armaments Center
Picatinny Arsenal, NJ 07801-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We present computational methods for positive kernel smoothing of piecewise linear data over uniform meshes. These methods or algorithms complete their work in an amount of time proportional to the amount of data present. The kernel used here is a B-spline which can be of arbitrarily high smoothness. The smoothed result or approximation may therefore also be as smooth as desired. The algorithms automatically evaluate the smooth approximation over any arbitrary mesh, including the original one if desired. Part of the reason why this smoothing may be done so efficiently stems from the fact that the kernel is never actually obtained or used explicitly. These methods lead naturally to consideration of smoothing the discrete cumulative distribution function corresponding to an ordered set of values of a random variable--a situation in which the original mesh is naturally always nonuniform. In this nonparametric estimation of a density, the use of a positive kernel is important, because the resulting integral smoothing operator is a monotone operator. In addition, derivatives of the smooth approximation may be obtained trivially.

14. SUBJECT TERMS
Kernel, Smoothing, Nonparametric, Window

15. NUMBER OF PAGES
29

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

INTRODUCTION

The problem we are concerned with in this report is developing a reliable and fast algorithm for continuous smoothing of piecewise linear (or piecewise constant) data defined over uniform or nonuniform meshes. The theory behind what we will do is covered in Reference 1. Here, we are concerned with actual computation. Often, large amounts of empirical data are obtained over uniform meshes and the smoothing is carried out over these same meshes. This case is the simplest one to consider, because we need not do any interpolation; but even in this simplest of cases, we must guard against algorithmic inefficiency when dealing with large amounts of data consisting of perhaps tens of thousands of points. The linear time property of an algorithm is therefore quite important in such cases and enables us to compute in seconds what might take hours otherwise. The situation becomes somewhat more complicated when the data is nonuniformly spaced and/or when we wish to evaluate the smoothed function over some other mesh. We use continuous positive kernel smoothing here, because of its shape (monotonicity) preserving properties and the ease with which we may interpolate functional and derivative values.

SMOOTHING BY REPEATED AVERAGING

In this section, we derive the smoothing formulas which will subsequently be applied. These formulas are discussed in Reference 1, but we include their derivation here for the sake of convenience. Consider the following smoothing operator S:

$$Sf(x) = \frac{1}{2h}\int_{x-h}^{x+h} f(t)dt$$

S operating on function f at point x is simply the average value of f over an interval of length $2h$ with x as the center of the interval. The parameter h is called the window parameter. This integral operator can also be written in kernel form as:

$$Sf(x) = \int_{-\infty}^{+\infty} f(t)K(t-x)dt$$

$$K(t) = \begin{cases} \frac{1}{2h} & \text{if } |t| < h \\ 0 & \text{if } |t| \geq h \end{cases}$$

Applying S a second time gives us

$$S^2 f(x) = \frac{1}{2h}\int_{x-h}^{x+h} Sf(t)dt$$

$$= \frac{1}{(2h)^2}\int_{x-h}^{x+h}\int_{t-h}^{t+h} f(u)dudt$$

Changing the order of integration in this double integral gives us

$$S^2 f(x) = \frac{1}{(2h)^2}\int_{x-2h}^{x}\int_{x-h}^{u+h} f(u)\,dt\,du$$

$$+\frac{1}{(2h)^2}\int_{x}^{x+2h}\int_{u-h}^{x+h} f(u)\,dt\,du$$

$$=\frac{1}{(2h)^2}\int_{x-2h}^{x} f(u)(u+2h-x)\,du$$

$$+\frac{1}{(2h)^2}\int_{x}^{x+2h} f(u)(x+2h-u)\,du$$

Writing this integral in kernel form, we have

$$S^2 f(x) = \int_{-\infty}^{+\infty} f(u)K(u-x)\,du$$

$$K(u) = \begin{cases} \dfrac{(2h+u)}{(2h)^2} & \text{if } -2h < u < 0 \\ \dfrac{(2h-u)}{(2h)^2} & \text{if } 0 \le u < 2h \\ 0 & \text{if } |u| \ge 2h \end{cases}$$

Note that the discontinuous kernel corresponding to S and the continuous kernel corresponding to S^2 are both nonnegative or essentially positive everywhere. Although we will not prove it here, successive applications of S result in a sequence of increasingly smoother and wider B-spline kernels of unit area. The width of the kernel corresponding to S^k is $2kh$. We now leave the subject of kernel form and proceed with the forms we actually use for computation.

Define the following set of integrals recursively:

$$I_0(x) = f(x)$$

$$I_{j+1}(x) = \int_a^x I_j(t)\,dt$$

where a is an arbitrary value in the domain of definition of f. Of course, the derivative of any member of this sequence is simply the preceding member of the sequence. This will make taking derivatives a trivial operation.

As before, applying S to f once, we have

$$Sf(x) = \frac{1}{2h}\int_{x-h}^{x+h} f(t)dt$$

$$= \frac{1}{2h}\left(\int_a^{x+h} f(t)dt - \int_a^{x-h} f(t)dt\right)$$

$$= \frac{1}{2h}(I_1(x+h) - I_1(x-h))$$

Applying S a second time and noting that S is a linear operator, we have

$$S^2 f(x) = \frac{1}{2h}(SI_1(x+h) - SI_1(x-h))$$

$$= \frac{1}{(2h)^2}\left(\int_{x-h}^{x+h} I_1(t+h)dt - \int_{x-h}^{x+h} I_1(t-h)dt\right)$$

$$= \frac{1}{(2h)^2}\left(\int_{x}^{x+2h} I_1(t)dt - \int_{x-2h}^{x} I_1(t)dt\right)$$

$$= \frac{1}{(2h)^2}(I_2(x+2h) - 2I_2(x) + I_2(x-2h))$$

Applying S a third time, we have

$$S^3 f(x) = \frac{1}{(2h)^2}(SI_2(x+2h) - 2SI_2(x) + SI_2(x-2h))$$

$$= \frac{1}{(2h)^3}\left(\int_{x-h}^{x+h} I_2(t+2h)dt - 2\int_{x-h}^{x+h} I_2(t)dt + \int_{x-h}^{x+h} I_2(t-2h)dt\right)$$

$$= \frac{1}{(2h)^3}(I_3(x+3h) - I_3(x+h) - 2(I_3(x+h) - I_3(x-h)) + I_3(x-h) - I_3(x-3h))$$

$$= \frac{1}{(2h)^3}(I_3(x+3h) - 3I_3(x+h) + 3I_3(x-h) - I_3(x-3h))$$

We can then derive the formula for the fourth smooth or, noting the appearance of the binomial coefficients with alternating signs, we can simply write down the result of applying S a fourth time:

$$S^4 f(x) = \frac{1}{(2h)^4}(I_4(x+4h) - 4I_4(x+2h) + 6I_4(x) - 4I_4(x-2h) + I_4(x-4h))$$

Also note that if we seek the derivative of this fourfold smoothed function, we simply reduce the indices of the I's by unity and get

$$\frac{d}{dx}S^4f(x)=\frac{1}{(2h)^4}(I_3(x+4h)-4I_3(x+2h)+6I_3(x)-4I_3(x-2h)+I_3(x-4h))$$

These ideas are elaborated upon in Reference 1.

COMPUTING THE INTEGRALS

We start with a continuous piecewise linear interpolation of the data.

$$f_i(x)=y_i+\frac{\Delta y_i}{\Delta x_i}(x-x_i) \qquad (x_i \le x \le x_{i+1})$$

For I_1, we have

$$I_1(x)=\int_{x_0}^{x} f(t)dt=\int_{x_0}^{x_i} f(t)dt+\int_{x_i}^{x} f_i(t)dt$$

$$=I_1(x_i)+\int_{x_i}^{x} y_i+\frac{\Delta y_i}{\Delta x_i}(t-x_i)dt$$

$$=I_1(x_i)+(x-x_i)y_i+\frac{1}{2}(x-x_i)^2\frac{\Delta y_i}{\Delta x_i}$$

$$I_1(x_{i+1})=I_1(x_i)+y_i\Delta x_i+\frac{1}{2}\Delta x_i\Delta y_i$$

$$=I_1(x_i)+\frac{1}{2}\Delta x_i(y_i+y_{i+1})$$

For I_2, we have

$$I_2(x) = \int_{x_0}^{x} I_1(t)dt = \int_{x_0}^{x_i} I_1(t)dt + \int_{x_i}^{x} I_1(t)dt$$

$$= I_2(x_i) + \int_{x_i}^{x} I_1(x_i) + (t-x_i)y_i + \frac{1}{2}(t-x_i)^2 \frac{\Delta y_i}{\Delta x_i} dt$$

$$= I_2(x_i) + (x-x_i)I_1(x_i) + \frac{1}{2}(x-x_i)^2 y_i + \frac{1}{6}(x-x_i)^3 \frac{\Delta y_i}{\Delta x_i}$$

$$I_2(x_{i+1}) = I_2(x_i) + I_1(x_i)\Delta x_i + \frac{1}{2}y_i \Delta x_i^2 + \frac{1}{6}\Delta y_i \Delta x_i^2$$

$$= I_2(x_i) + I_1(x_i)\Delta x_i + \Delta x_i^2 \left(\frac{1}{3}y_i + \frac{1}{6}y_{i+1} \right)$$

For I_3, we have

$$I_3(x) = \int_{x_0}^{x} I_2(t)dt = \int_{x_0}^{x_i} I_2(t)dt + \int_{x_i}^{x} I_2(t)dt$$

$$= I_3(x_i) + \int_{x_i}^{x} I_2(x_i) + (t-x_i)I_1(x_i) + \frac{1}{2}(t-x_i)^2 y_i + \frac{1}{6}(t-x_i)^3 \frac{\Delta y_i}{\Delta x_i} dt$$

$$= I_3(x_i) + (x-x_i)I_2(x_i) + \frac{1}{2}(x-x_i)^2 I_1(x_i) + \frac{1}{6}(x-x_i)^3 y_i + \frac{1}{24}(x-x_i)^4 \frac{\Delta y_i}{\Delta x_i}$$

$$I_3(x_{i+1}) = I_3(x_i) + I_2(x_i)\Delta x_i + \frac{1}{2}I_1(x_i)\Delta x_i^2 + \frac{1}{6}y_i \Delta x_i^3 + \frac{1}{24}\Delta y_i \Delta x_i^3$$

$$= I_3(x_i) + I_2(x_i)\Delta x_i + \frac{1}{2}I_1(x_i)\Delta x_i^2 + \Delta x_i^3 \left(\frac{1}{8}y_i + \frac{1}{24}y_{i+1} \right)$$

This is as far in the sequence of *I*'s as we will go in this report. The previous set of formulas gives us two things. The first thing is the recursion relationships we must use to compute the nodal values of the first three integrals. The second thing is the formulas for interpolating the integrals at arbitrary points. The nodal integral values are essentially computed only once and the amount of work involved is obviously proportional to the amount of data present. The interpolations may obviously be done in constant time, once the intervals of the various arguments have been located. Since, in the evaluation of the smoothed function, we march through the data only once as we locate the intervals of points in the desired mesh, this component of the algorithm will also be done in linear time. The amount of work done to evaluate the smooth approximation will therefore be k_1n+k_2m arithmetic operations, where n is the amount of data and m is the number of points at which we wish to evaluate the smoothed function. It is also important to note that the amount of work involved in this computational scheme is completely independent of the size of the window parameter *h*.

This scheme does have a weakness, however. If the nodal values of the integrals are computed for an entire very large set of data, roundoff error in the integral computation and in the necessary interpolations (involving subtractions) will completely destroy the computation, especially for relatively small window sizes. The results will be so contaminated with error they will meaningless. For a small to intermediate amount of data, however, the results will be very good. It is therefore wise to periodically compute the nodal values of the integrals in smaller clumps as we march through the data. Remember that the lower limit of integration for the integrals is analytically arbitrary. A good rule of thumb is to compute the integrals over some multiple of the total window width. This modification naturally forces us to do a bit of extra computation. Each clump of integral values must overlap the previous clump by at least the total window width.

Therefore, with a window multiplier of ten, we will do only about ten percent more computational work. Sufficient accuracy can still be attained with window multipliers as large as one hundred. In any case, the modified algorithm can handle any amount of data in linear time with sufficient accuracy. An involved error analysis could perhaps be done for this algorithm, but it is more practical to evaluate its accuracy by applying the algorithm to an artificial set of data obtained from a straight line, since the smoothing operators leave such data undisturbed analytically but not numerically. All we need do is to compare the original data with that produced by the algorithm with its inherent roundoff error. Since the smoothing formulas require data on either side of x and the domain of the data is finite, the smoothing operator cannot be used near the ends of the data. For this case, we merely use the simple derivative estimating properties of the smoothing formulas to expand the smoothed function in a Taylor series at a point sufficiently far from the ends and compute the smoothed function near the ends via these Taylor series. Finally, if the data has been taken over a uniform mesh (equally spaced values of the independent variable) and the smoothed function is desired over this same mesh, only the nodal values of the integrals are necessary and no interpolations need be done.

NONPARAMETRIC DENSITY ESTIMATION

The simplest and most common example of nonparametric density estimation is the frequency histogram. Our goal in this section is to improve upon the discontinuous histogram by replacing it with a density estimate that is smooth to the extent of being twice continuously differentiable. Attainment of this goal will give us a density that is not only more aesthetically pleasing to the eye, but one with which we can make more accurate probability statements and calculate more accurate percentiles. This is of course done via positive kernel smoothing. Assuming that a sequence of values of a random variable x has been sorted in ascending order, we can define the discrete cumulative distribution function as

$$F_n(x)=\begin{cases} 0 & \text{if } x<x_0 \\ \dfrac{i+1}{n} & \text{if } x_i<x<x_{i+1} \\ 1 & \text{if } x>x_{n-1} \end{cases}$$

Note that this function is defined on the entire real line, so that we need not use Taylor series to smooth near the ends of a finite domain. Also, it does not matter how the distribution function is defined at the data points or whether any of the x values are repeated values. Our first estimate of the density will be the derivative of the fourth smooth of F_n. Hence, we need three successive integrals of F_n. Since the values of x are always nonuniformly spaced, interpolations are mandatory.

For I_1, we have

$$I_1(x)=\int_{x_0}^{x} F_n(t)dt=\int_{x_0}^{x_i} F_n(t)dt+\int_{x_i}^{x}\frac{i+1}{n}dt$$

$$=I_1(x_i)+\frac{i+1}{n}(x-x_i)$$

$$I_1(x_{i+1})=I_1(x_i)+\frac{(i+1)\Delta x_i}{n}$$

For I_2, we have

$$I_2(x)=\int_{x_0}^{x} I_1(t)dt=\int_{x_0}^{x_i} I_1(t)dt+\int_{x_i}^{x} I_1(x_i)+\frac{i+1}{n}(t-x_i)dt$$

$$=I_2(x_i)+I_1(x_i)(x-x_i)+\frac{i+1}{2n}(x-x_i)^2$$

$$I_2(x_{i+1})=I_2(x_i)+I_1(x_i)\Delta x_i+\frac{(i+1)\Delta x_i^2}{2n}$$

For I_3, we have

$$I_3(x)=\int_{x_0}^{x} I_2(t)dt=\int_{x_0}^{x_i} I_2(t)dt+\int_{x_i}^{x} I_2(x_i)+I_1(x_i)(t-x_i)+\frac{i+1}{2n}(t-x_i)^2dt$$

$$=I_3(x_i)+I_2(x_i)(x-x_i)+\frac{1}{2}I_1(x_i)(x-x_i)^2+\frac{i+1}{6n}(x-x_i)^3$$

$$I_3(x_{i+1})=I_3(x_i)+I_2(x_i)\Delta x_i+\frac{1}{2}I_1(x_i)\Delta x_i^2+\frac{(i+1)\Delta x_i^3}{6n}$$

Density f is given by

$$f(x)=\frac{1}{(2h)^4}[I_3(x+4h)-4I_3(x+2h)+6I_3(x)-4I_3(x-2h)+I_3(x-4h)]$$

Note that although we have derived the interpolation formulas for I_1 and I_2 in passing, only the interpolation formula for I_3 is necessary since we have no need of Taylor series expansions. Also, the formula for $I_3(x)$ can be used to extrapolate beyond the last point.

PRESERVING THE VARIANCE

The sample mean computed from the data will coincide with the mean computed from the density estimate. The sample variance computed from the data will not coincide with the variance computed from the density estimate. The density variance will always exceed the sample variance by an ever larger amount as the window parameter is increased. The object of this section is to derive a simple transformation which will keep the density variance the same as the sample variance for all values of the window parameter h. We now return to the use of kernel form momentarily. The continuous cumulative distribution function estimate is given by

$$F(x)=S^4F_n(x)=\int_{-\infty}^{\infty}F_n(t)K(t-x)dt$$

Integrating by parts, we have

$$F(x)=F_n(\infty)\int_{-\infty}^{\infty}K(t-x)dt-\int_{-\infty}^{\infty}\int_{-\infty}^{t}K(u-x)du\,dF_n(t)$$

$$=1-\frac{1}{n}\sum_{i=0}^{n-1}\int_{-\infty}^{x_i}K(u-x)du$$

Differentiating the cumulative distribution function gives us the kernel form of the density:

$$F'(x)=f(x)=\frac{1}{n}\sum_{i=0}^{n-1}\int_{-\infty}^{x_i}K'(u-x)du=\frac{1}{n}\sum_{i=0}^{n-1}K(x_i-x)$$

This is the common form for a nonparametric density estimator based on kernel K. However, it is exactly the wrong form to use for computation. If it were used to evaluate a density at each of n points, an amount of work proportional to the square of the amount of data would be necessary. That is, instead of a linear time algorithm, we would have a quadratic time algorithm. We can, however, use this formula to determine the relationship between the variance computed from the density, the sample variance, and the window parameter h. Consider the expected value (via f) of an arbitrary function p.

$$E[p(x)] = \int_{-\infty}^{\infty} p(x) f(x) dx = \int_{-\infty}^{\infty} p(x) \frac{1}{n} \sum_{i=0}^{n-1} K(x - x_i) dx$$

$$= \frac{1}{n} \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} p(x + x_i) K(x) dx$$

With $p(x)=x$, we have

$$E[x] = \mu_f = \frac{1}{n} \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} (x + x_i) K(x) dx = \frac{1}{n} \sum_{i=0}^{n-1} x_i = \hat{\mu}$$

since $xK(x)$ is an odd function.

We therefore see that the sample mean and the density computed mean are the same. Computing the variance (again via f), we have

$$V[x] = \sigma_f^2 = E[(x - \mu_f)^2] = \frac{1}{n} \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} (x + x_i - \mu_f)^2 K(x) dx$$

$$= \frac{1}{n} \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} (x^2 + 2x(x_i - \mu_f) + (x_i - \mu_f)^2) K(x) dx$$

$$= \int_{-\infty}^{\infty} x^2 K(x) dx + \frac{1}{n} \sum_{i=0}^{n-1} (x_i - \mu_f)^2$$

$$= S^4 x^2(0) + \hat{\sigma}^2$$

But S^4x^2 is given by

$$S^4 x^2 = \frac{1}{(2h)^4} (I_4(x + 4h) - 4I_4(x + 2h) + 6I_4(x) - 4I_4(x - 2h) + I_4(x - 4h))$$

$$I_4(x) = \frac{x^6}{2^3 \cdot 3^2 \cdot 5}$$

Hence

$$S^4x^2(0)=\frac{1}{(2h)^4}(2I_4(4h)-8I_4(2h))$$

$$=\frac{1}{(2h)^4}\left(\frac{2^{13}\cdot h^6}{2^3\cdot 3^2\cdot 5}-\frac{2^9\cdot h^6}{2^3\cdot 3^2\cdot 5}\right)$$

$$=\frac{h^2}{3^2\cdot 5}(2^6-2^2)=\frac{4h^2}{3}$$

We therefore finally have the relation between the density-computed variance, the sample variance, and the window parameter:

$$\sigma_f^2=\hat{\sigma}^2+\frac{4h^2}{3}$$

Now let g be the density of a new random variable (x').

$$g(x)=\alpha f(\mu_f+\alpha(x-\mu_f))$$

This transformation represents a simultaneous compression (about the mean) of f in the x direction and a stretching of f in the y direction. Our objective now is to compute the transformation parameter α in terms of the sample variance and the window parameter such that the variance of the new random variable computed via g will be the sample variance.

Considering again the expected value (via g) of an arbitrary function p of the new random variable, we have

$$E[p(x')]=\int_{-\infty}^{\infty}p(x)g(x)dx$$

$$=\int_{-\infty}^{\infty}p(x)\alpha f(\mu_f+\alpha(x-\mu_f))dx$$

$$=\int_{-\infty}^{\infty}p(x+\mu_f)\alpha f(\mu_f+\alpha x)dx$$

$$=\int_{-\infty}^{\infty}p\left(\frac{x}{\alpha}+\mu_f\right)f(\mu_f+x)dx$$

$$=\int_{-\infty}^{\infty}p\left(\frac{x-\mu_f}{\alpha}+\mu_f\right)f(x)dx$$

Letting $p(x)=1$, we see that g is indeed a legitimate density. Letting $p(x)=x$, we see that the means for f and g are the same. Computing the variance of x', we have

$$p(x)=(x-\mu_f)^2$$

$$\sigma_g^2=\frac{\sigma_f^2}{\alpha^2}=\hat{\sigma}^2$$

Therefore,

$$\alpha=\sqrt{\frac{\sigma_f^2}{\hat{\sigma}^2}}=\sqrt{1+\frac{4h^2}{3\hat{\sigma}^2}}$$

Now, since

$$g(x)=\alpha f(\mu_f+\alpha(x-\mu_f))$$

or equivalently

$$g\left(\frac{x-\mu_f}{\alpha}+\mu_f\right)=\alpha f(x)$$

we compute f over some x mesh and transform the density data according to

$$f\leftarrow\alpha f,\ x\leftarrow\frac{x-\hat{\mu}}{\alpha}+\hat{\mu}$$

in order to obtain the g density data that preserves both the sample mean and the sample variance.

OPTIMAL WINDOW PARAMETER

In this section we develop some machinery for selecting the optimal window parameter h for a nonparametric density estimator. As before, we call the density estimator g and we will call the actual density γ. The mean square error (MSE) for a given value of x is given by

$$\begin{aligned}MSE(x)&=E[(g(x)-\gamma(x))^2]=E[(g(x)-E[g(x)]+E[g(x)]-\gamma(x))^2]\\&=E[(g(x)-E[g(x)])^2\\&\quad+2(g(x)-E[g(x)])(E[g(x)]-\gamma(x))\\&\quad+(E[g(x)]-\gamma(x))^2]\\&=E[(g(x)-E[g(x)])^2]+(E[g(x)]-\gamma(x))^2\\&=V[g(x)]+B(x)^2\end{aligned}$$

$V[g(x)]$ is the variance of g and $B(x)$ is the bias. The integral of the mean square error (IMSE) is given by

$$IMSE = \int_{-\infty}^{+\infty} V[g(x)] + B(x)^2 dx$$

Our goal is to approximately minimize IMSE with respect to the window parameter h. We seek a formula that is asymptotically valid for large n and small h. Recalling that

$$g(x) = \alpha f(\mu + \alpha(x-\mu))$$

and

$$f(x) = \frac{1}{n}\sum_{i=0}^{n-1} K(x_i - x)$$

where K is the kernel resulting from four applications of the smoothing operator S, we have

$$g(x) = \frac{\alpha}{n}\sum_{i=0}^{n-1} K(x_i - \mu - \alpha(x-\mu))$$

But

$$x_i - \mu - \alpha(x-\mu) = x_i - x + x - \mu - \alpha(x-\mu) = x_i - x + (1-\alpha)(x-\mu)$$

and

$$\alpha = \sqrt{1 + \frac{4h^2}{3\sigma^2}} \sim 1 + \frac{2h^2}{3\sigma^2} \quad (h \to 0)$$

We therefore have

$$x_i - \mu - \alpha(x-\mu) \sim x_i - x - h^2\beta(x)$$

where

$$\beta(x) = \frac{2(x-\mu)}{3\sigma^2}$$

Hence

$$g(x) = \frac{\alpha}{n}\sum_{i=0}^{n-1} K(x_i - x - h^2\beta)$$

Since the x's are identically distributed independent random variables, the expected value and variance of g (via γ) are given by

$$E[g(x)] = \frac{\alpha}{n}\sum_{i=0}^{n-1} E[K(x_i - x - h^2\beta)]$$

$$= \alpha E[K(x_i - x - h^2\beta)]$$

$$V[g(x)] = \frac{\alpha^2}{n^2}\sum_{i=0}^{n-1} V[K(x_i - x - h^2\beta)]$$

$$= \frac{\alpha^2}{n} V[K(x_i - x - h^2\beta)]$$

We now begin to obtain simple asymptotic expressions for the expectation and variance of these kernel expressions. Now K is a kernel which becomes a Dirac delta as h approaches zero, so we take note of the fact that

$$hK(hu) = k(u), \quad K(u) = \frac{1}{h}k\left(\frac{u}{h}\right)$$

where k is a standard kernel independent of h. In our case, the support of k is (-4,4). First, we have

$$E[K(x_i - x - h^2\beta)] = \int_{-\infty}^{+\infty} K(u - x - h^2\beta)\gamma(u)du$$

$$= \int_{-\infty}^{+\infty} K(u)\gamma(u + x + h^2\beta)du = \int_{-\infty}^{+\infty} K(hu)\gamma(x + hu + h^2\beta)hdu$$

$$= \int_{-\infty}^{+\infty} k(u)\gamma(x + hu + h^2\beta)du$$

Expanding part of the integrand in a Taylor series, we have

$$\gamma(x + hu + h^2\beta) = \gamma(x) + \gamma'(x)(hu + h^2\beta)$$

$$+\frac{1}{2}\gamma''(x)(hu + h^2\beta)^2 + \frac{1}{6}\gamma'''(x)(hu + h^2\beta)^3 + O(h^4)$$

$$= \gamma(x) + \gamma'(x)(hu + h^2\beta)$$

$$+\frac{1}{2}\gamma''(x)(h^2u^2 + 2h^3u\beta + h^4\beta^2)$$

$$+\frac{1}{6}\gamma'''(x)(h^3u^3 + 3h^4u^2\beta + 3h^5u\beta^2 + h^6\beta^3) + O(h^4)$$

Since k is an even function, the terms involving odd powers of u make no contribution to the integral. The contributing terms of the series are therefore

$$\gamma_c(x+hu+h^2\beta) = \gamma(x)+h^2\beta\gamma'(x)+\frac{1}{2}h^2u^2\gamma''(x)+O(h^4)$$

Hence,

$$E[K(x_i-x-h^2\beta)] = \int_{-\infty}^{+\infty}k(u)\left(\gamma(x)+h^2\beta\gamma'(x)+\frac{1}{2}h^2u^2\gamma''(x)+O(h^4)\right)du$$
$$= \gamma(x)+h^2\beta\gamma'(x)+\frac{1}{2}h^2\gamma''(x)\int_{-\infty}^{+\infty}u^2k(u)du+O(h^4)$$

But

$$\int_{-\infty}^{+\infty}u^2k(u)du = \int_{-\infty}^{+\infty}\left(\frac{u}{h}\right)^2k\left(\frac{u}{h}\right)\frac{1}{h}du$$
$$= \frac{1}{h^2}\int_{-\infty}^{+\infty}u^2K(u)du = \frac{4}{3}$$

Therefore

$$E[K(x_i-x-h^2\beta)] = \gamma(x)+\frac{2h^2}{3\sigma^2}(x-\mu)\gamma'(x)+\frac{2}{3}h^2\gamma''(x)+O(h^4)$$
$$= \gamma(x)+\frac{2h^2}{3\sigma^2}((x-\mu)\gamma'(x)+\sigma^2\gamma''(x))+O(h^4)$$

The bias is therefore given approximately by

$$B(x) \sim \alpha\left(\gamma(x)+\frac{2h^2}{3\sigma^2}((x-\mu)\gamma'(x)+\sigma^2\gamma''(x))\right)-\gamma(x)$$
$$=(\alpha-1)\gamma(x)+\frac{2h^2\alpha}{3\sigma^2}((x-\mu)\gamma'(x)+\sigma^2\gamma''(x))$$
$$\sim \frac{2h^2}{3\sigma^2}(\gamma(x)+(x-\mu)\gamma'(x)+\sigma^2\gamma''(x))$$
$$= \frac{2h^2}{3\sigma^2}b(x)$$

In order to obtain the variance of g, we must first compute

$$E[K(x_i-x-h^2\beta)^2]=\int_{-\infty}^{+\infty}K(u-x-h^2\beta)^2\gamma(u)du=\int_{-\infty}^{+\infty}K(u)^2\gamma(u+x+h^2\beta)du$$
$$=\int_{-\infty}^{+\infty}K(hu)^2\gamma(x+hu+h^2\beta)hdu=\frac{1}{h}\int_{-\infty}^{+\infty}k(u)^2\gamma(x+hu+h^2\beta)du$$
$$=\frac{\gamma(x)}{h}\int_{-\infty}^{+\infty}k(u)^2du+O(h)=\frac{c\gamma(x)}{h}+O(h)$$

Now, since

$$V[K(x_i-x-h^2\beta)]=E[K(x_i-x-h^2\beta)^2]-E[K(x_i-x-h^2\beta)]^2$$

we have the desired variance of g

$$V[g(x)]=\frac{\alpha^2}{n}\left(\frac{c\gamma(x)}{h}-\gamma(x)^2+O(h)\right)$$
$$\sim\frac{c\gamma(x)}{nh}-\frac{\gamma(x)^2}{n}$$

The integral of the mean square error is therefore

$$IMSE\sim\frac{c}{nh}-\frac{1}{n}\int_{-\infty}^{+\infty}\gamma(x)^2dx+\frac{4h^4}{9\sigma^4}\int_{-\infty}^{+\infty}b(x)^2dx$$

Letting

$$J=\int_{-\infty}^{+\infty}b(x)^2dx$$

and setting the derivative with respect to h of the integral of the mean square error equal to zero, we have

$$-\frac{c}{nh^2}+\frac{16h^3J}{9\sigma^4}=0$$

Solving for h, we have

$$h_{opt}\sim\left(\frac{9c\sigma^4}{16nJ}\right)^{\frac{1}{5}}$$

We therefore see that as the sample size becomes large, the optimal window parameter becomes small, albeit very slowly. We now simplify the integral J. First, we assume that

$$\lim_{x\to\pm\infty}x\gamma(x)=0=\lim_{x\to\pm\infty}x\gamma'(x)$$

Now,

$$\begin{aligned} b(x)^2 &= \left(\gamma(x)+(x-\mu)\gamma'(x)+\sigma^2\gamma''(x)\right)^2 \\ &= \gamma(x)^2+(x-\mu)^2\gamma'(x)^2+\sigma^4\gamma''(x)^2 \\ &\quad +2(x-\mu)\gamma(x)\gamma'(x) \\ &\quad +2\sigma^2\gamma(x)\gamma''(x) \\ &\quad +2\sigma^2(x-\mu)\gamma'(x)\gamma''(x) \end{aligned}$$

Integrating by parts, we have

$$\begin{aligned} J_1 &= \int_{-\infty}^{+\infty}(x-\mu)\gamma(x)\gamma'(x)dx = \int_{-\infty}^{+\infty}(x-\mu)\gamma(x)d\gamma(x) \\ &= -\int_{-\infty}^{+\infty}\gamma(x)\left(\gamma(x)+(x-\mu)\gamma'(x)\right)dx = -\int_{-\infty}^{+\infty}\gamma(x)^2dx - J_1 \\ J_1 &= -\frac{1}{2}\int_{-\infty}^{+\infty}\gamma(x)^2dx \end{aligned}$$

and

$$J_2 = \int_{-\infty}^{+\infty}\gamma(x)\gamma''(x)dx = \int_{-\infty}^{+\infty}\gamma(x)d\gamma'(x) = -\int_{-\infty}^{+\infty}\gamma'(x)^2dx$$

and

$$\begin{aligned} J_3 &= \int_{-\infty}^{+\infty}(x-\mu)\gamma'(x)\gamma''(x)dx = \int_{-\infty}^{+\infty}(x-\mu)\gamma'(x)d\gamma'(x) \\ &= -\int_{-\infty}^{+\infty}\gamma'(x)\left(\gamma'(x)+(x-\mu)\gamma''(x)\right)dx = -\int_{-\infty}^{+\infty}\gamma'(x)^2dx - J_3 \\ J_3 &= -\frac{1}{2}\int_{-\infty}^{+\infty}\gamma'(x)^2dx \end{aligned}$$

from which we get

$$\begin{aligned} J &= \int_{-\infty}^{+\infty}\gamma(x)^2+(x-\mu)^2\gamma'(x)^2+\sigma^4\gamma''(x)^2dx+2\left(-\frac{1}{2}\int_{-\infty}^{+\infty}\gamma(x)^2dx\right) \\ &\quad +2\sigma^2\left(-\int_{-\infty}^{+\infty}\gamma'(x)^2dx\right)+2\sigma^2\left(-\frac{1}{2}\int_{-\infty}^{+\infty}\gamma'(x)^2dx\right) \\ &= \int_{-\infty}^{+\infty}\left((x-\mu)^2-3\sigma^2\right)\gamma'(x)^2+\sigma^4\gamma''(x)^2dx \end{aligned}$$

We can get a clearer idea of the dependence of J on σ by invoking the standardized version of γ. Let Γ be the standardized version with zero mean and unity variance. First, we have

$$\frac{1}{\sigma}\Gamma\left(\frac{x-\mu}{\sigma}\right) = \gamma(x)$$

$$\frac{1}{\sigma^2}\Gamma'\left(\frac{x-\mu}{\sigma}\right)=\gamma'(x)$$

$$\frac{1}{\sigma^3}\Gamma''\left(\frac{x-\mu}{\sigma}\right)=\gamma''(x)$$

Hence,

$$J=\int_{-\infty}^{+\infty}((x-\mu)^2-3\sigma^2)\frac{1}{\sigma^4}\Gamma'\left(\frac{x-\mu}{\sigma}\right)^2+\frac{1}{\sigma^2}\Gamma''\left(\frac{x-\mu}{\sigma}\right)^2dx$$
$$=\int_{-\infty}^{+\infty}(x^2-3\sigma^2)\frac{1}{\sigma^4}\Gamma'\left(\frac{x}{\sigma}\right)^2+\frac{1}{\sigma^2}\Gamma''\left(\frac{x}{\sigma}\right)^2dx$$
$$=\frac{1}{\sigma}\int_{-\infty}^{+\infty}(x^2-3)\Gamma'(x)^2+\Gamma''(x)^2dx$$

where the integral factor is completely independent of σ. We therefore have

$$h_{opt}=\sigma\left(\frac{9c}{16nJ\sigma}\right)^{\frac{1}{5}}$$

We will now find an expression for optimal h for a particular Γ. A candidate for this particular case must naturally be twice differentiable. The function k naturally comes to mind as a good unimodal candidate, having zero mean. It does not, however, have unity variance and must therefore be properly scaled. Consider

$$\Gamma(x)=ak(ax)$$
$$\int_{-\infty}^{+\infty}x^2\Gamma(x)dx=1=\int_{-\infty}^{+\infty}x^2ak(ax)dx=\frac{1}{a^2}\int_{-\infty}^{+\infty}x^2k(x)dx=\frac{4}{3a^2}$$

Hence,

$$a^2=\frac{4}{3}$$

Also,

$$\Gamma'(x)=a^2k'(ax),\quad \Gamma''(x)=a^3k''(ax)$$

Computing the $J\sigma$ integral, we have

$$J\sigma = \int_{-\infty}^{+\infty} (x^2-3)a^4k'(ax)^2 + a^6k''(ax)^2 dx$$
$$= a^3 \int_{-\infty}^{+\infty} \left(\frac{x^2}{a^2} - 3\right) k'(x)^2 + a^2k''(x)^2 dx$$
$$= \frac{2}{9\sqrt{3}} \int_{-\infty}^{+\infty} (9x^2-36)k'(x)^2 + 16k''(x)^2 dx$$

Therefore, optimal h for this particular standardized density is given by

$$h_{opt} = \sigma \left(\frac{81\sqrt{3}\int_0^{\infty} k(x)^2 dx}{32n\int_0^{\infty} (9x^2-36)k'(x)^2 + 16k''(x)^2 dx} \right)^{\frac{1}{5}}$$

From Reference 1, we have

$$k(x) = \begin{cases} \dfrac{(4-x)^3 - 4(2-x)^3}{96} & \text{if } 0 \le x \le 2 \\ \dfrac{(4-x)^3}{96} & \text{if } 2 \le x \le 4 \\ 0 & \text{if } x > 4 \\ k(-x) & \text{if } x < 0 \end{cases}$$

from which the first and second derivatives of k can be obtained. Then, we can compute the approximation

$$h_{opt} \sim \sigma \left(\frac{22}{n}\right)^{\frac{1}{5}}$$

This asymptotic approximation can be used as a nominal estimate of optimal h for a unimodal density. For multimodal data, or in order to pick out more of the variations present in the data, one can naturally use smaller values of h, but we at least have a value to start with.

Since this report is mainly computation oriented, we include some relevant C routines in the Appendix. The first part of the Appendix is devoted to the C preprocessor commands that describe the syntax used in the routines. The second, third, and fourth parts of the Appendix are devoted to uniform mesh smoothing, nonparametric density estimation, and nonuniform mesh smoothing, respectively.

REFERENCE

1. Royce Soanes, "Function Smoothing by Repeated Averaging," U.S. Army ARDEC Technical Report ARCCB-TR-88012, Benét Laboratories, Watervliet, NY, March 1988.

APPENDIX

```
#define If if(
#define Then ) {
#define Elsif } else if(
#define Else } else {
#define fI };
#define iF (
#define theN ) ?
#define elsE :
#define Fi ;
#define And &&
#define Or ||
#define Do for(;;) {
#define oD }
#define Un if(
#define nU ) break;
#define As if(!(
#define sA )) break;
#define NL printf("\n");
#define REAL(r) printf(#r "=%le ",r);
#define INT(i) printf(#i "=%ld ",i);
#define REALS(n,r) { long _i_=0; \
                     printf("\n"); \
                     Do As _i_ < n sA \
                         printf(#r "[%ld]=%le ",_i_,r[_i_]); _i_++; oD \
                     printf("\n"); };
#define INTS(n,i) { long _j_=0; \
                    printf("\n"); \
                    Do As _j_ < n sA \
                      printf(#i "[%ld]=%ld ",_j_,i[_j_]); _j_++; oD \
                    printf("\n"); };
#define AVER(bool,fname) if(!(bool)) \
                           { printf("\n" #bool " is false in " #fname); \
                             exit(0); };
#define DENY(bool,fname) if(bool) \
                           { printf("\n" #bool " in " #fname); \
                             exit(0); };
                                                    .
                                                    .
                                                    .

#include "syntax.h"

#define DIM 500

double I1[DIM],I2[DIM],I3[DIM];

void pkints(m,M,y)
        long m,M;
```

```
            double *y;
{ long i,im1;
  i=m+1; I1[m]=0.0; I2[m]=0.0; I3[m]=0.0;
  Do Un i > M nU
     im1=i-1;
     I1[i]=I1[im1]+(y[im1]+y[i])/2.0;
     I2[i]=I2[im1]+(I1[im1]+(y[im1]/3.0+y[i]/6.0));
     I3[i]=I3[im1]+(I2[im1]+(I1[im1]/2.0+(y[im1]/8.0+y[i]/24.0)));
     i++; oD }


void pksmooth(n,y,w,wm,s)
              long n,w,wm;
              double *y,*s;
{ void pkints(long,long,double*);
  double H,f0,f1,f2,x;
  long nint,m,M,i,imw,ipw,im3w,ip3w,j,ilast;
  DENY(n > DIM,pksmooth)
  DENY(6*w+1 > n,pksmooth)
  DENY(wm < 2,pksmooth)
  nint=(6*w+1)*wm;
  m=0; M=m+nint; If M >= n Then M=n-1; fI
  pkints(m,M,y);
  H=2.0*(double)w; H=H*H*H;
  i=3*w;
  imw=i-w; ipw=i+w;
  im3w=i-3*w; ip3w=i+3*w;
  f0=(I3[ip3w]-I3[im3w]-3.0*(I3[ipw]-I3[imw]))/H;
  f1=(I2[ip3w]-I2[im3w]-3.0*(I2[ipw]-I2[imw]))/H;
  f2=(I1[ip3w]-I1[im3w]-3.0*(I1[ipw]-I1[imw]))/H;
  j=0;
  Do Un j > i nU
     x=(double)(j-i);
     s[j]=f0+x*(f1+x*f2/2.0);
     j++; oD
  i=j; ilast=n-1-3*w;
  Do imw=i-w; ipw=i+w;
     im3w=i-3*w; ip3w=i+3*w;
     If ip3w > M
          Then m=im3w; M=m+nint;
          If M >= n Then M=n-1; fI
          pkints(m,M,y); fI
     f0=(I3[ip3w]-I3[im3w]-3.0*(I3[ipw]-I3[imw]))/H;
     s[i]=f0;
     i++; Un i > ilast nU oD
  f1=(I2[ip3w]-I2[im3w]-3.0*(I2[ipw]-I2[imw]))/H;
  f2=(I1[ip3w]-I1[im3w]-3.0*(I1[ipw]-I1[imw]))/H;
  j=i; i=ilast;
```

```
  Do As j < n sA
    x=(double)(j-i);
    s[j]=f0+x*(f1+x*f2/2.0);
    j++; oD }
```

```
#include "syntax.h"
#include "math.h"

#define DIM 500

double I1[DIM],I2[DIM],I3[DIM];

void cdfints(n,x,m,M)
             long n,m,M;
             double *x;
{ long i,im1;
  double dx,rn,ri;
  DENY(n > DIM,cdfints)
  If m < 0 Then m=0; fI
  If M >= n Then M=n-1; fI
  DENY(m >= M,cdfints)
  i=m+1; I1[m]=0.0; I2[m]=0.0; I3[m]=0.0; rn=(double)n;
  Do Un i > M nU
    im1=i-1; dx=x[i]-x[im1];
    ri=(double)i;
    I1[i]=I1[im1]+dx*ri/rn;
    I2[i]=I2[im1]+dx*(I1[im1]+dx*ri/(2.0*rn));
    I3[i]=I3[im1]+dx*(I2[im1]+dx*(I1[im1]/2.0+dx*ri/(6.0*rn)));
    i++; oD }


double I3c(n,x,a,ila)
           long n,ila;
           double *x,a;
{ double dx,rn,rila;
  If a < x[0] Then return (0.0); fI
  DENY(a < x[ila],I3c)
  dx=a-x[ila]; rn=(double)n; rila=(double)ila;
  return (I3[ila]+dx*(I2[ila]+dx*(I1[ila]/2.0+dx*(rila+1.0)/(6.0*rn)))); }


void npden(n,x,h,wm,nd,xd,yd)
           long n,wm,nd;
           double *x,h,*xd,*yd;
{ void cdfints(long,double*,long,long);
```

```
  double I3c(long,double*,double,long);
  void musig(long,double*,double*,double*);
  double xm,xl,xr,xll,xrr,xrrr,H,E1,E2,mu,sig,alpha;
  long id,im,il,ir,ill,irr,irrr,nm1,i;
  DENY(n > DIM,npden)
  id=0; im=0; il=0; ir=0; ill=0; irr=0; nm1=n-1;
  H=2.0*(double)h; H=H*H*H*H;
  id=0; irrr=0;
  Do As id < nd sA
    xm=xd[id];
    xl=xd[id]-2.0*h; xr=xd[id]+2.0*h;
    xll=xd[id]-4.0*h; xrr=xd[id]+4.0*h;
    Do Un im == nm1 nU Un x[im+1] > xm nU im++; oD
    Do Un il == nm1 nU Un x[il+1] > xl nU il++; oD
    Do Un ir == nm1 nU Un x[ir+1] > xr nU ir++; oD
    Do Un ill == nm1 nU Un x[ill+1] > xll nU ill++; oD
    Do Un irr == nm1 nU Un x[irr+1] > xrr nU irr++; oD
    If xrr > x[irrr] And irrr < nm1
        Then xrrr=xll+8.0*wm*h;
            Do Un irrr == nm1 nU Un x[irrr] > xrrr nU irrr++; oD
            cdfints(n,x,ill,irrr); fI
    E1=I3c(n,x,xrr,irr)+6.0*I3c(n,x,xm,im)+I3c(n,x,xll,ill);
    E2=I3c(n,x,xr,ir)+I3c(n,x,xl,il);
    yd[id]=(E1-4.0*E2)/H;
    If yd[id] < 0.0 Then yd[id]=0.0; fI
    id++; oD
  musig(n,x,&mu,&sig);
  alpha=h/sig; alpha=sqrt(1.0+4.0*alpha*alpha/3.0);
  i=0;
  Do As i < nd sA
    xd[i]=mu+(xd[i]-mu)/alpha;
    yd[i]=alpha*yd[i];
    i++; oD }


void musig(n,x,mu,sig)
          long n;
          double *x,*mu,*sig;
{ long i;
  double s,d;
  i=0; s=0.0; Do As i < n sA s+=x[i]; i++; oD
  *mu=s/(double)n;
  i=0; s=0.0; Do As i < n sA d=x[i]-(*mu); s+=d*d; i++; oD
  *sig=sqrt(s/(double)n); }


void sift(heap,root,last)
        double *heap;
```

```
        long root,last;
{ long l,r;
  double t;
  Do l=2*root+1;
    If l > last Then return; fI
    r=l+1;
    If r > last
        Then If heap[root] < heap[l]
                Then t=heap[root]; heap[root]=heap[l]; heap[l]=t; fI
             return; fI
    If heap[root] >= heap[l] And heap[root] >= heap[r]
      Then return; fI
    If heap[l] > heap[r]
        Then t=heap[l]; heap[l]=heap[root]; heap[root]=t; root=l;
        Else t=heap[r]; heap[r]=heap[root]; heap[root]=t; root=r; fI    oD }


void makeheap(array,last)
            double *array;
            long last;
{ long i1,i2,i;
  void sift(double*,long,long);
  i1=0;
  Do i1=2*i1+1; Un i1 > last nU oD
  i1=(i1-1)/2;
  Do i2=i1-1; i1=i2/2;
    If last/2 < i2 Then i2=last/2; fI
    i=i1;
    Do Un i > i2 nU
        sift(array,i,last); i++; oD
    Un i1 == 0 nU oD }


void heapsort(n,x)
            long n;
            double *x;
{ long last,i;
  double t;
  void makeheap(double*,long),sift(double*,long,long);
  last=n-1; makeheap(x,last);
  Do Un last == 0 nU
    t=x[0]; x[0]=x[last]; x[last]=t; last--;
    sift(x,0,last); oD
  i=1;
  Do As i < n sA
    DENY(x[i-1] > x[i],heapsort) i++; oD }
```

```
#include "syntax.h"

#define DIM 500

double I1[DIM],I2[DIM],I3[DIM];

void pkints(n,x,y,m,M)
            long n,m,M;
            double *x,*y;
{ long i,im1;
  double dx;
  DENY(n > DIM,pkints)
  If m < 0 Then m=0; fI
  If M >= n Then M=n-1; fI
  DENY(m >= M,pkints)
  i=m+1; I1[m]=0.0; I2[m]=0.0; I3[m]=0.0;
  Do Un i > M nU
     im1=i-1; dx=x[i]-x[im1];
     I1[i]=I1[im1]+dx*(y[im1]+y[i])/2.0;
     I2[i]=I2[im1]+dx*(I1[im1]+dx*(y[im1]/3.0+y[i]/6.0));
     I3[i]=I3[im1]+dx*(I2[im1]+dx*(I1[im1]/2.0+dx*
     (y[im1]/8.0+y[i]/24.0)));
     i++; oD }


double I1c(n,x,y,a,ila)
            long n,ila;
            double *x,*y,a;
{ double dx,r,E;
  If ila < 0 Then ila=0; fI
  If ila > n-2 Then ila=n-2; fI
  dx=a-x[ila]; r=dx/(x[ila+1]-x[ila]);
  E=((2.0-r)*y[ila]+r*y[ila+1])/2.0;
  return (I1[ila]+dx*E); }


double I2c(n,x,y,a,ila)
            long n,ila;
            double *x,*y,a;
{ double dx,r,E;
  If ila < 0 Then ila=0; fI
  If ila > n-2 Then ila=n-2; fI
  dx=a-x[ila]; r=dx/(x[ila+1]-x[ila]);
  E=((3.0-r)*y[ila]+r*y[ila+1])/6.0;
  return (I2[ila]+dx*(I1[ila]+dx*E)); }
```

```
double I3c(n,x,y,a,ila)
        long n,ila;
        double *x,*y,a;
{ double dx,r,E;
  If ila < 0 Then ila=0; fI
  If ila > n-2 Then ila=n-2; fI
  dx=a-x[ila]; r=dx/(x[ila+1]-x[ila]);
  E=((4.0-r)*y[ila]+r*y[ila+1])/24.0;
  return (I3[ila]+dx*(I2[ila]+dx*(I1[ila]/2.0+dx*E))); }


void nupks(n,x,y,w,wm,ns,xs,ys)
        long n,wm,ns;
        double *x,*y,w,*xs,*ys;
{ void pkints(long,double*,double*,long,long);
  double I1c(long,double*,double*,double,long);
  double I2c(long,double*,double*,double,long);
  double I3c(long,double*,double*,double,long);
  double xl,xr,xll,xrr,xrrr,H,E1,E2,f0,f1,f2,xx;
  long is,il,ir,ill,irr,irrr,j;
  DENY(n > DIM,nupks)
  is=0; il=0; ir=1; ill=0; irr=1;
  Do Un xs[is]-3.0*w >= x[0] nU is++; oD
  xl=xs[is]-w; xr=xs[is]+w; xll=xs[is]-3.0*w; xrr=xs[is]+3.0*w;
  DENY(xrr > x[n-1],nupks)
  Do Un x[il+1] > xl nU il++; oD
  Do Un x[ir] >= xr nU ir++; oD
  Do Un x[ill+1] > xll nU ill++; oD
  Do Un x[irr] >= xrr nU irr++; oD
  irrr=irr; xrrr=xll+6.0*wm*w;
  If xrrr > x[n-1] Then xrrr=x[n-1]; fI
  Do Un x[irrr] >= xrrr nU irrr++; oD
  pkints(n,x,y,ill,irrr);
  H=2.0*(double)w; H=H*H*H;
  E1=I3c(n,x,y,xrr,irr-1)-I3c(n,x,y,xll,ill);
  E2=I3c(n,x,y,xr,ir-1)-I3c(n,x,y,xl,il);
  f0=(E1-3.0*E2)/H;
  E1=I2c(n,x,y,xrr,irr-1)-I2c(n,x,y,xll,ill);
  E2=I2c(n,x,y,xr,ir-1)-I2c(n,x,y,xl,il);
  f1=(E1-3.0*E2)/H;
  E1=I1c(n,x,y,xrr,irr-1)-I1c(n,x,y,xll,ill);
  E2=I1c(n,x,y,xr,ir-1)-I1c(n,x,y,xl,il);
  f2=(E1-3.0*E2)/H;
  j=0;
  Do As j < is sA
    xx=xs[j]-xs[is];
    ys[j]=f0+xx*(f1+xx*f2/2.0);
    j++; oD
```

```
Do xl=xs[is]-w; xr=xs[is]+w; xll=xs[is]-3.0*w; xrr=xs[is]+3.0*w;
   DENY(xrr > x[n-1],nupks)
   Do Un x[il+1] > xl nU il++; oD
   Do Un x[ir] >= xr nU ir++; oD
   Do Un x[ill+1] > xll nU ill++; oD
   Do Un x[irr] >= xrr nU irr++; oD
   If xrr > x[irrr]
      Then xrrr=xll+6.0*wm*w;
      If xrrr > x[n-1] Then xrrr=x[n-1]; fI
      Do Un x[irrr] >= xrrr nU irrr++; oD
      pkints(n,x,y,ill,irrr); fI
   E1=I3c(n,x,y,xrr,irr-1)-I3c(n,x,y,xll,ill);
   E2=I3c(n,x,y,xr,ir-1)-I3c(n,x,y,xl,il);
   f0=(E1-3.0*E2)/H;
   ys[is]=f0;
   Un xs[is+1]+3.0*w > x[n-1] nU is++; oD
E1=I2c(n,x,y,xrr,irr-1)-I2c(n,x,y,xll,ill);
E2=I2c(n,x,y,xr,ir-1)-I2c(n,x,y,xl,il);
f1=(E1-3.0*E2)/H;
E1=I1c(n,x,y,xrr,irr-1)-I1c(n,x,y,xll,ill);
E2=I1c(n,x,y,xr,ir-1)-I1c(n,x,y,xl,il);
f2=(E1-3.0*E2)/H;
j=is+1;
Do As j < ns sA
   xx=xs[j]-xs[is];
   ys[j]=f0+xx*(f1+xx*f2/2.0);
   j++; oD }
```

TECHNICAL REPORT INTERNAL DISTRIBUTION LIST

	NO. OF COPIES
CHIEF, DEVELOPMENT ENGINEERING DIVISION	
ATTN: AMSTA-AR-CCB-DA	1
-DB	1
-DC	1
-DD	1
-DE	1
CHIEF, ENGINEERING DIVISION	
ATTN: AMSTA-AR-CCB-E	1
-EA	1
-EB	1
-EC	
CHIEF, TECHNOLOGY DIVISION	
ATTN: AMSTA-AR-CCB-T	2
-TA	1
-TB	1
-TC	1
TECHNICAL LIBRARY	
ATTN: AMSTA-AR-CCB-O	5
TECHNICAL PUBLICATIONS & EDITING SECTION	
ATTN: AMSTA-AR-CCB-O	3
OPERATIONS DIRECTORATE	
ATTN: SMCWV-ODP-P	1
DIRECTOR, PROCUREMENT & CONTRACTING DIRECTORATE	
ATTN: SMCWV-PP	1
DIRECTOR, PRODUCT ASSURANCE & TEST DIRECTORATE	
ATTN: SMCWV-QA	1

NOTE: PLEASE NOTIFY DIRECTOR, BENÉT LABORATORIES, ATTN: AMSTA-AR-CCB-O OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST

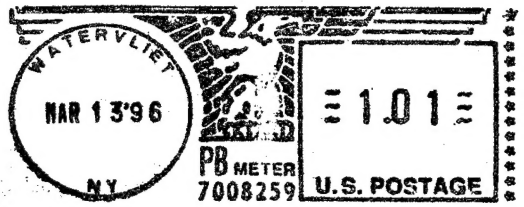
	NO. OF COPIES
ASST SEC OF THE ARMY RESEARCH AND DEVELOPMENT ATTN: DEPT FOR SCI AND TECH THE PENTAGON WASHINGTON, D.C. 20310-0103	1
ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER ATTN: DTIC-OCP (ACQUISITION GROUP) BLDG. 5, CAMERON STATION ALEXANDRIA, VA 22304-6145	2
COMMANDER U.S. ARMY ARDEC ATTN: SMCAR-AEE	1
SMCAR-AES, BLDG. 321	1
SMCAR-AET-O, BLDG. 351N	1
SMCAR-FSA	1
SMCAR-FSM-E	1
SMCAR-FSS-D, BLDG. 94	1
SMCAR-IMI-I, (STINFO) BLDG. 59 PICATINNY ARSENAL, NJ 07806-5000	2
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-DD-T, BLDG. 305 ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-WT-PD (DR. B. BURNS) ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. MATERIEL SYSTEMS ANALYSIS ACTV ATTN: AMXSY-MP ABERDEEN PROVING GROUND, MD 21005-5071	1
COMMANDER ROCK ISLAND ARSENAL ATTN: SMCRI-ENM ROCK ISLAND, IL 61299-5000	1
MIAC/CINDAS PURDUE UNIVERSITY P.O. BOX 2634 WEST LAFAYETTE, IN 47906	1
COMMANDER U.S. ARMY TANK-AUTMV R&D COMMAND ATTN: AMSTA-DDL (TECH LIBRARY) WARREN, MI 48397-5000	1
COMMANDER U.S. MILITARY ACADEMY ATTN: DEPARTMENT OF MECHANICS WEST POINT, NY 10966-1792	1
U.S. ARMY MISSILE COMMAND REDSTONE SCIENTIFIC INFO CENTER ATTN: DOCUMENTS SECTION, BLDG. 4484 REDSTONE ARSENAL, AL 35898-5241	2
COMMANDER U.S. ARMY FOREIGN SCI & TECH CENTER ATTN: DRXST-SD 220 7TH STREET, N.E. CHARLOTTESVILLE, VA 22901	1
COMMANDER U.S. ARMY LABCOM MATERIALS TECHNOLOGY LABORATORY ATTN: SLCMT-IML (TECH LIBRARY) WATERTOWN, MA 02172-0001	2
COMMANDER U.S. ARMY LABCOM, ISA ATTN: SLCIS-IM-TL 2800 POWER MILL ROAD ADELPHI, MD 20783-1145	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, BENÉT LABORATORIES, CCAC, U.S. ARMY TANK-AUTOMOTIVE AND ARMAMENTS COMMAND, AMSTA-AR-CCB-O, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST (CONT'D)

	NO. OF COPIES
COMMANDER U.S. ARMY RESEARCH OFFICE ATTN: CHIEF, IPO P.O. BOX 12211 RESEARCH TRIANGLE PARK, NC 27709-2211	1
DIRECTOR U.S. NAVAL RESEARCH LABORATORY ATTN: MATERIALS SCI & TECH DIV CODE 26-27 (DOC LIBRARY) WASHINGTON, D.C. 20375	1 1
WRIGHT LABORATORY ARMAMENT DIRECTORATE ATTN: WL/MNM EGLIN AFB, FL 32542-6810	1
WRIGHT LABORATORY ARMAMENT DIRECTORATE ATTN: WL/MNMF EGLIN AFB, FL 32542-6810	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, BENÉT LABORATORIES, CCAC, U.S. ARMY TANK-AUTOMOTIVE AND ARMAMENTS COMMAND, AMSTA-AR-CCB-O, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.



DEPARTMENT OF THE ARMY
ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER
BENÉT LABORATORIES, CCAC
US ARMY TANK-AUTOMOTIVE AND ARMAMENTS COMMAND
WATERVLIET, N.Y. 12189-4050

OFFICIAL BUSINESS
AMSTA-AR-CCB-O
TECHNICAL LIBRARY

DEPARTMENT OF THE ARMY

DA Label 18-1, Sep 83
Edition of Oct 74 will be used until exhausted.

OFFICIAL BUSINESS

DEFENSE TECHNICAL INFO CENTER
ATTN: DTIC-OCP (ACQUISITIONS)
8725 JOHN J. KINGMAN ROAD
STE 0944
FT. BELVOIR, VA 22060-6218